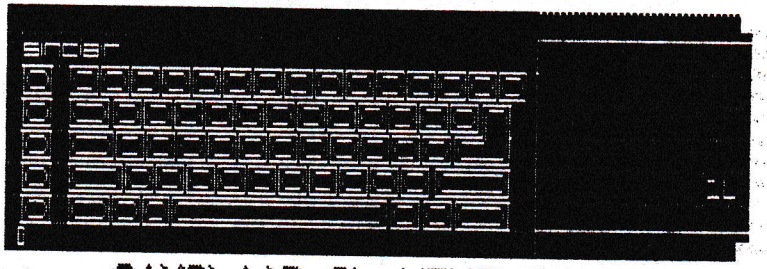
QL_DOC



SINCLAIR QL NEWSLETTER

No. 2 FEVRIER-MARS 1988 $ 1.25

:: REMark ::

Hé oui, voici le deuxième numéro de QL_DOC, le seul newsletter FRANCOPHONE en Amérique du nord consacré uniquement au QL! Il faut du courage pour vouloir accomplir la tâche de publier REGULIEREMENT un bulletin d'informations pour le QL. Nous seulement QL_DOC ne s'intéresse qu'au QL, qui est, il faut se l'avouer, pas l'ordinateur le plus répandu par les temps qui court, mais il est écrit en FRANCAIS ce qui rend "presque" impossible la pénération chez les QLs hors-QUEBEC et les QLs américains. Mais enfin, si on est marginal, on est marginal jusqu'au bout (de souffle !?!).

Dans ce numéro, on trouve un article qui décrit la commande BEEP du SUPERBASIC. Egalement, une explication de l'organisation de la mémoire-écran du QL. Et enfin, un logiciel pour imprimer de jolis étiquettes pour vos disquettes bourrées de petits programmes.

N'hésitez pas à envoyez du matériel à QL_DOC, car vous possédez peut-être la réponse à une question qu'un autre lecteur se pose...

A la prochaine,

Real-d

QL_DOC est réalisé avec l'aide du logiciel FRONT PAGE EXTRA de GAP SOFTWARE.
L'impression se fait lentement mais surement sur imprimante STAR NX-10.
Rédacteur: Réal Gagnon , 4870 Henri-Julien , Montréal Québec , CANADA H2T 2E1
QL_DOC parait au 2 mois. L'abonnement coûte $ 10.00 (6 nos , 1 an).
Vos commentaires, questions, articles, programmes sont appréciés!

INPUT <-> OUTPUT

(Le courrier des lecteurs)

Les lecteurs de QL_DOC sont sans doute au courant que le QL version américaine est équipé d'une ROM JSU (PRINT VER$).Ils ont sans doute vite découvert aussi, comme moi, que mis a part la possibilité de connecter le QL à une TV du type employée en Amérique du Nord, la ROM JSU n'apportait que des problèmes de compatibilités avec beaucoup de programmes (SIGN DESIGNER, CADPAK, GRAPHIQL, FRONT PAGE EXTRA?, ETC...).Il existe une solution (drastique?) à ce problème: changer de ROM. En effet SHARP's fournit une ROM qui remplace les deux ROMs du QL en version JSU ou JS. L'avantage de changer JSU -> JSU est que la consommation de courant du QL diminue car il n'y a plus qu'une ROM qui consomme (+- 20%). En plus de cet avantage, si vous changez pour une ROM JS, votre ordinateur devient compatible avec tout ce qui se produit de logiciels en Angleterre.Cependant, un nouveau problème pointe à l'horizon: vos propres programmes que vous avez créés sur ROM JSU sont maintenant incompatibles avec votre ROM JS! En fait, j'exagère un peu: seulement ceux faisant appel à des commandes graphiques. En effet le problème de compatibilité provient de la commande SCALE, ou plus exactement du ratio de l'axe des y à l'axe des x employé par cette commande et qui diffère d'une ROM à l'autre.Bref, si vous voulez éviter d'avoir à retaper des sections complètes de vos programmes, utiliser le truc suivant: faites varier les pixels x de vos commandes graphiques (POINT, LINE...) à l'aide d'une variable, disons "v". Il suffit de savoir la valeur à donner à "v": pour passer de JSU à JS, cette valeur est 1.1778. Donc, par exemple, la commande LINE x,y TO x1,y1 devient LINE x*v,y TO x1*v,y1.Par contre si vous avez une TV et que vous êtes obligé de conserver la ROM JSU, vous aurez souvent des problèmes quand vous tapez des listings de revues anglaises. Employez le même truc, mais cette fois donnez la valeur .8490 à "v".J'espère avoir contribué à jeter un peu de lumière sur un problème qui m'a fait sacr...
Jean-Claude Touzin, c.p. 41, La Sarre, P.Q. J9Z 2X4

Voici une soultion intéressante à un sacré problème. J'ai eu l'occasion de voir un de ces bidules chez **GAGNON ELECTRONIQUE**, mais celle-ci venait de **CURRY COMP.** Il y avait bien la ROM JS mais en plus sur le même EPROM le **SUPER TOOLKIT II.** Habituellement, les ROMs du QL se présente sous la forme de 27128 (32Kx8bits), mais sur ce machin nous retrouvons un 27512 (64Kx8bits), ce qui explique pourquoi nous avons autant de place disponible. Quant à l'économie d'énergie dont Jean-Claude fait mention, elle s'explique du fait que c'est la version CMOS du 27512 qui est utilisée, le 27C512.

Real-d

L'ECRAN du SINCLAIR QL

L'écran du QL occupe 32 K-octets de la mémoire RAM. C'est en partie pour cette raison que sur un QL standard de 128K, il n'y a que 80K disponible à la programmation, une partie étant réservée pour les variables systèmes et à la mémoire-écran.

La mémoire-écran débute à l'adresse 131072 décimale (20000 hex) et progresse pendant 32K en mot de 16 bits jusqu'à l'adresse 163840 (28000 hex).

*Pour sauver le contenu d'un écran, nous pouvons faire en SUPERBASIC

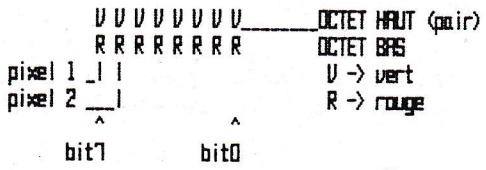
```
SBYTES xxxx_dessin_pic,131072,32768
ou
SBYTES xxxx_dessin_pic,2^17,2^15
```

Contrairement au 2068/SPRECTRUM, l'écran du QL est continu, c'est-à-dire qu'elle débute dans le coin supérieur gauche et se termine dans le coin inférieur droit.

Il y a deux modes graphiques, 512x256(4 couleurs) et 256x256(8 couleurs).

1- MODE 512x256

Ce mode graphique offre une résolution de 512 pixels horizontales et 256 verticales, avec une possibilité de 4 couleurs. Chaque mot de 16 bits détermine la couleur de 8 pixels à l'écran.



.Composition d'un mot (mode 512x256)

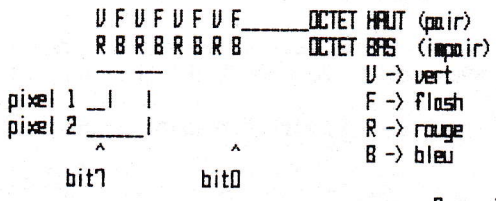
Ainsi si nous voulons le 4ième pixel rouge, la bit 5 de l'octet haut doit être à "0" et celle de l'octet bas à "1".Voici les combinaisons possibles pour les 4 couleurs.

COULEUR	BIT "1"
noir	aucune
rouge	rouge
vert	vert
blanc	rouge+vert

2- MODE 256x256

Ce mode donne une résolution de 256 pixels horizontales et 256 verticales. Il y a maintenant 8 couleurs disponibles. Un mot de 16 bits de la mémoire-écran affecte 4 pixels.

L'écran du SINCLAIR QL (suite ...)



.Composition d'un mot (mode 256x256)

Voici les combinaisons possibles pour les 8 couleurs.

COULEUR	BIT "1"
noir	aucune
bleu	bleu
rouge	rouge
vert	vert
magenta	bleu+rouge
cyan	bleu+vert
jaune	vert+rouge
blanc	bleu+vert+rouge

NOTE: Pour chaque pixel, on met la bit FLASH à "1" si on veut qu'il flash! Donc chaque mot contrôle 4 pixels. Si nous voulons le premier pixel bleu, le second rouge, le troisième magenta et le quatrième vert, nous aurons:

Octet haut	00	00	00	10	-> 2 décimale
Octet bas	01	10	11	00	-> 108 décimale
	bl	ro	ma	ve	

*Pour joindre les 2 octets, on procède de la façon suivante:

Multiplie l'octet haut par 256 -> 2 * 256 = 512
Additionne le résultat avec l'octet bas -> 512 + 108 = 620

Faites POKE_W 2^17,620 pour vérifier les 4 premiers pixels de la mémoire-écran. N'oubliez pas il faut être en mode TV!

3-APPLICATION

Comme application, vous une fonction qui est l'équivalent la fonction POINT du 2068/SPECTRUM. Elle retourne la couleur d'un point à l'écran.

- Il y a une version pour chaque mode vidéo.
- La procédure prend en considération l'écran en entier, la référence étant le coin inférieur gauche (0,0). Ici les WINDOWs ne sont pas pris en considération.

```
*Fonction pour MODE 256x256

DEFine FuNction color_l (x,y)                    <- x,y sont les coordonnées
  LOCal col,mem,haut,bas
  mem = 163712 + (x DIV 8)* 2-y*128              <- trouve l'adresse
  haut=PEEK (mem) DIV (4^(3-(x DIV 2)MOD 4))     <- sépare les 2 bits
                                                    correspondant au pixel
  bas=PEEK (mem+1) DIV (4^(3-x DIV 2)MOD 4))     <- vérifie bit BLEU & ROUGE
  col=bas && 3
  col=col+(haut && 2) *2                         <- vérifie bit VERT
  col=col+(haut && 1) *4                         <- vérifie bit FLASH
  RETurn col                                     <- retourne la couleur
END DEFine

Les valeurs possibles sont : 0 NOIR     1 BLEU
                             2 ROUGE    3 MAGENTA
                             4 VERT     5 CYAN
                             6 JAUNE    7 BLANC

*Fonction pour MODE 512x256

DEFine FuNction color_h (x,y)                    <- x,y sont les coordonnées
   LOCal col,mem,haut,bas
   mem=163712+(x DIV 8)*2-y*128                  <- trouve l'adresse
   haut=PEEK (mem) DIV (2^(7-(x MOD 8)))         <- sépare le bit qui représente le
   bas=PEEK (mem+1) DIV (2^(7-(x MOD 8)))           pixel
   col= bas && 1                                 <- vérifie le bit ROUGE
   col= col+(haut && 1)*2                        <- vérifie le bit VERT
   RETurn col*2                                  <- retourne la couleur
END DEFine

Les valeurs possibles sont :     0 NOIR    2 ROUGE
                                 4 VERT    6 BLANC
```

On utilise la fonction de la façon suivante:

-si on veut vérifier le pixel à la position 100,50
nous faisons couleur=col_h(100,50)
si nous sommes en mode 512x256, sinon nous faisons
couleur=col_l(100,50)
pour le mode 256x256.

DLQ_BAS

DLQ_BAS est un programme qui vous permet de faire de jolies étiquettes pour vos disquettes. En principe, le programme fonctionne avec un QL avec de FLPs mais il peut être facilement modifié pour fonctionner seulement avec des µdrives. Le problème avec ceux-çi est qu'ils sont trop petits pour pouvoir coller une étiquette! Mais enfin, pour la modification, changez les "FLP1" pour des "MDV1" aux lignes suivantes: 185,330,340,350,360,570,750.

Pour l'impression, votre imprimante doit être EPSON compatible. Pour mettre le plus d'informations possibles sur l'étiquette, nous mettons l'imprimante en mode ELITE, COMPRESSE, SUPERSCRIPT, avec un interligne de 16/216".

Voici les codes de contrôle nécessaires:

ELITE -> ESC "M"
COMPRESSE -> CHR$(15)
SUPERSCRIPT -> ESC "S0"
INTERLIGNE -> ESC "3" CHR$(16)

Vérifiez si votre votre imprimante supporte ces codes...

Pour démarrer le programme faites LRUN xxxx_DLQ_BAS.

La première question est pour déterminer sur quel "DEVICE" vous voulez faire un "DIRECTORY". Si on fait l'étiquette de la disquette présente dans FLP1_, on écrit FLP1_.

Puis on peut mettre un commentaire ou un titre, sinon on fait simplement ENTER.

Le DIRECTORY sera alors classé en ordre alphabétique. Pour imprimer l'étiquette, on fait LISTE_PRT.

NOTE:
Les procédures SORT et QSORT, ont pour source le magazine SINCLAIR QL WORLD.

.exemple d'étiquette

GAMES#2
453/720 sectors

BATNAV1_SET	BATNAV2_SET	BATNAV_BOT	batnav	CARTES_SET	CHIFLET_BAS
MASTERMIND_	MEMORY_BAS	MEMORY_BOT	MORPION_BAS	MORP_CASE_D	MORP_TABL_D
NIM	NIM_BOT	PENDU_BAS	QLOVIS_BAS	QLOVIS_BOT	QLOVIS_SET
YAHTZEE_BAS					

```
100 REMark disk label organizer .5  par GAG-oB6 MTL
110 :
120     WINDOW 512,256,0,0
130     BORDER 3,45
140     PAPER 0:INK 7
150     CLS
160     PRINT "IMPORTANT: La disquette dans FLP1_ ne doit pas être protégée"
170     PRINT "           en ecriture."\\
180     INPUT '"Directory" de quel "Device" (ex. flp1_ mdv1_) ?  ';dev$
185     IF dev$='':dev$='flp1_'
190     INPUT 'Un titre général ou commentaire ? ';titre$
200      read_dir             :REMark lecture du directory
210      display_files        :REMark montre les fichiers
220      sort fi$             :REMark classé en ordre alphabetique
230      display_files        :REMark montre les fichiers en ordre
240      BEEP 1000,10
250      CSIZE#0,1,0
260      BORDER#0,1,7
270      PRINT#0,'FAITES   LISTE_PRT  -> pour imprimer'
280      CSIZE#0,0,0
290      STOP
300 :
310 :
320 DEFine PROCedure read_dir
330      DELETE flp1_dir1
340      OPEN_NEW#5,flp1_dir1
350      DIR#5,dev$
360      CLOSE#5:OPEN#5,flp1_dir1
370 :
380 REMark trouve l'espace prise sur disque
390 :
400     CLS
410     INPUT#5,fr_sp$
420     PRINT fr_sp$
430 :
440 REMark compte les fichiers et trouve le plus long
450 :
460       INPUT#5,dummy$   :REMark espace vide
470       nfiles=0
480       lmax=0
490       REPeat loop
500          INPUT #5,a$
510          IF EOF(#5):EXIT loop
520          lfi=LEN(a$)
530          IF lfi>lmax:lmax=lfi
540          nfiles=nfiles+1
550       END REPeat loop
560       PRINT 'il y a ';nfiles;' fichiers sur cette disquette!'
```

```
570       CLOSE#5:OPEN#5,flp1_dir1
580       INPUT#5,dummy$:INPUT#5,dummy$   :REMark espace disponible + vide
590 :
600 REMark creation du tableau contenant les fichiers...
610 :
620       DIM fi$(nfiles,lmax)
630 :
640 REMark lecture des fichiers
650 :
660    i=1
670  REPeat loop
680    INPUT#5,a$
690    IF EOF(#5):EXIT loop:END IF
700    IF a$<>'dir1'
710    fi$(i)=a$
720    i=i+1
730    END IF
740  END REPeat loop
750    CLOSE#5:DELETE flp1_dir1
760 END DEFine
770 :
780 :
790 DEFine PROCedure display_files
800     f=1
810     PRINT
820 FOR i=1 TO INT(nfiles/5)
830     FOR ii=0,15,30,45,60
840        PRINT TO ii,fi$(f);
850        f=f+1
860     END FOR ii
870   PRINT
880 END FOR i
890    l=0
900 FOR i=f TO nfiles
910    PRINT TO l,fi$(i);
920    l=l+15
930 END FOR i
940 END DEFine
950 :
960 :
970 REMark procedures sort & qsort, source SINCLAIR QL WORLD!
980 :
990 DEFine PROCedure sort (array)
1000   PRINT \\
1010   INK 4
1020   PRINT 'Je fais le classement...'\\
1030   INK 7
```

```
1040    qsort array,1,DIMN(array)
1050 END DEFine sort
1060 :
1070 DEFine PROCedure qsort (array,bottom,top)
1080    LOCal loop,lo,hi,pointer
1090    lo=bottom:hi=top
1100    pointer=bottom
1110 REPeat loop
1120    IF lo>=hi:EXIT loop
1130    IF array(lo)>array(hi)
1140       temp$=array(lo)
1150       array(lo)=array(hi)
1160       array(hi)=temp$
1170         IF pointer=lo
1180            lo=lo+1:pointer=hi
1190         ELSE
1200            hi=hi-1:pointer=lo
1210         END IF
1220    ELSE
1230       IF pointer=lo
1240            hi=hi-1
1250       ELSE
1260            lo=lo+1
1270       END IF
1280   END IF
1290 END REPeat loop
1300   IF ABS(top-bottom)<2 THEN RETurn
1310    qsort array, bottom, pointer-1
1320    qsort array, pointer+1, top
1330 END DEFine qsort
1340 :
1350 :
1360 DEFine PROCedure liste_prt
1370    l$='                '
1380    OPEN#5,ser1
1390 :
1400 REMark imprimante ->ELITE,COMPRESSE,SUPERSCRIPT,interligne 16/216"
1410 :
1420    PRINT#5,CHR$(27);"M";CHR$(15);CHR$(27);"S0";CHR$(27);"3";CHR$(16);
1430    PRINT#5,CHR$(27);'W1';titre$\fr_sp$;CHR$(27);'W0' :REMark large on/off
1440    nbr_col=6
1450    n_lignes=INT(nfiles/nbr_col)
1460    FOR ligne=0 TO n_lignes
1470        FOR colonne=1 TO nbr_col
1480         IF ((ligne*nbr_col)+colonne)>nfiles:EXIT ligne
1490            long_f=LEN(fi$((ligne*nbr_col)+colonne))
1500            IF long_f>12
1505               f$=fi$(((ligne*nbr_col)+colonne),1 TO 11)
```

```
1510              ELSE
1520              f$=fi$((ligne*nbr_col)+colonne)
1530            END IF
1540            l_f=13-LEN(f$)
1550          PRINT #5,f$;l$(1 TO l_f);
1560        END FOR colonne
1570        PRINT#5
1580    END FOR ligne
1590    PRINT#5,CHR$(27);"@"\\\\\\\\\\\\\:CLOSE#5
1600 END DEFine
1610 :
1620 DEFine PROCedure update
1630 DELETE flp1_dlo_bas
1640 SAVE flp1_dlo_bas
1650 END DEFine
```

BEEP BEEP yeah!...

La syntaxe de la commande BEEP est:

BEEP durée,pitch_a,pitch_b,grad_x,grad_y,wrap,fuzz,random

1* DUREE
Valeurs possibles : 0 à 32767

Chaque unité représente 1/72 de seconde, la valeur 32767 représente 2.36 secondes et 0 signifie que le son répété à l'infini.

2* PITCH_A
Valeurs possibles : 0 à 255

C'est le son proprement dit. Mais le problème est que le son produit est en complexe relation avec sa durée. Il n'est pas question de dire que la note FA sera toujours la même valeur. Un PITCH de 1 est bas et un de 255 est haut.

3* PITCH_B
Valeurs possibles : 0 à 255

C'est le son final vers où le PITCH va se rendre. Mais avant d'aller plus loin, voyons les 2 autres paramètres!

4* GRAD_X
Valeurs possibles : -32768 à -32767

Indique le temps de transition que le son prend pour passer de PITCH_A à PITCH_B, en augmentant (ou diminuant) de GRAD_Y.

5* GRAD_Y
Valeurs possibles : -8 à 7

Indique de combien on va augmenter à chaque transition pour atteindre PITCH_B.

EXEMPLE: BEEP 0,10,20,100,2

0 → le son se répète à l'infini
10 → c'est le son initial
20 → c'est le son final
100 → c'est le temps de transition, 100% 1/72 de seconde
2 → on augmente de 2 après chaque transition

.ici on commence à 10
.on attend 100 unités
.on augmente de 2, c'est-à-dire à 12
.on attend 100 unités
.on augmente de 2, c'est-à-dire à 14
.et on continue jusqu'au PITCH égale à 20
.après on recommence, mais dans le sens inverse en soustrayant 2, jusqu'à 10.

6° WRAP
Valeurs possibles : 0 → 15

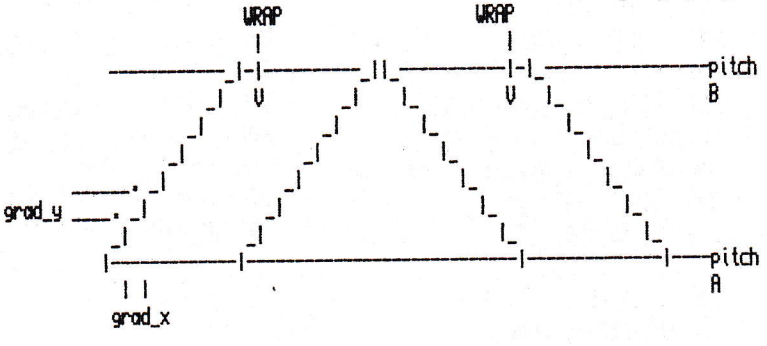
Comme nous avons vu précedemment, on part de PITCH_A, on monte jusqu'à PITCH_B pour redescendre par la suite vers PITCH_A. WRAP peut faire changer tout ça!

Si WRAP égale à 1, nous aurons la situation suivante.

. nous partons de PITCH_A, on monte jusqu'à PITCH_B.
. à PITCH_B, on recommence à PITCH_A pour remonter vers PITCH_B, c'est le WRAP.
. une fois rendu à PITCH_B, on redescend vers PITCH_A comme
. auparavant. mais voilà lorsque rendu à PITCH_A, on remonte
. à PITCH_B sans attendre pour redescendre vers PITCH_A, pour un WRAP. Est-ce clair ?!?

note: si WRAP égale 15, on WRAP tout le temps.

.exemple graphique



7* FUZZ
Valeurs possibles : 0 + 15
8* RANDOM
Valeurs possibles : 0 + 15

FUZZ ajoute un élément aléatoire au PITCH.
Tandis que RANDOM ajoute un élément aléatoire au GRAD_Y.

EXEMPLE DE L'UTILISATION DE LA COMMANDE BEEP

RAYON DE LA MORT	BEEP 0,255,1,200,-1,5,0,9
ALERTE GENERALE	BEEP 0,255,1,200,-1,5
AUTO DE COURSE	BEEP 0,100,250,1000,-1,2,10,8
POLICE (Francaise)	BEEP 0,30,37,9000,7,0,0,0
PIECE ALEATOIRE	BEEP 0,1,1,4500,0,5,0,9
REVEILLE MATIN	BEEP 0,0,67,475
LASER	BEEP 0,0,12,43,1,0,0,0

Pour mettre le son a OFF, on fait simplement BEEP avec aucun paramètre!

EFFETS SPECIAUX

```
100 REMark effets speciaux
110 MODE 8
120 WINDOW 512,256,0,0:PAPER 0:INK 0
130 CLS
140 REPeat loop
150  WINDOW 250,200,150,10:BORDER 1,7
160  PAPER 2:CLS
170  PRINT \\\\
180  CSIZE 1,0
190  PRINT "DEMONSTRATION "\"de "
200  PRINT "l'utilisation"\"de la"
210  PRINT "commande RECOL"
220  CSIZE 0,0
230  PRINT \\\"Appuyez une touche!"
235  CSIZE 2,1
240  PAUSE 100
250   fx 0,7*8
260   fx 7*8,0
270 END REPeat loop
280 :
290 DEFine PROCedure fx(a,b)
300 IF a<b:pas=1:ELSE pas=-1
310   FOR i=a TO b STEP pas
320   BORDER i-(b AND pas=-1),i+1
330   c=i MOD 7
331 REMark    BEEP 0,100,250,1,8
332 :
340   RECOL c,c,c,c,c,c,c,c
345 :
347   BEEP
350   END FOR i
360 END DEFine
```